

# Project Documentation



**VIT**<sup>®</sup>  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **AN AUTOMATED PREDICTION MODEL FOR DIABETIC RETINOPATHY USING CNN**

**By**

**Ravulapalli Sai Krishna (21BRS1628)**

**Sadineni Varun Kumar(21BRS1540)**

**Bandi Lokeshvardhan (21BAI1654)**

**TEAM-592538**

<b>S.NO</b>	<b>Title of Figure</b>	<b>Page No</b>
<b>1</b>	INTRODUCTION	03
<b>2</b>	LITERATURE SURVEY	05
<b>3</b>	IDEATION & PROPOSED SOLUTION	08
<b>4</b>	REQUIREMENT ANALYSIS	09
<b>5</b>	PROJECT DESIGN	11
<b>6</b>	PROJECT PLANNING & SCHEDULING	20
<b>7</b>	CODING & SOLUTIONING	24
<b>8</b>	RESULTS	33
<b>9</b>	TESTING	37
<b>10</b>	ADVANTAGES & DISADVANTAGES	40
<b>11</b>	CONCLUSION	41
<b>12</b>	FUTURE SCOPE	42
<b>13</b>	APPENDIX	43

## **Introduction**

Diabetic retinopathy is a condition that may do in people who have diabetes. It causes progressive damage to the retina, the light-sensitive filling at the reverse of the eye. Diabetic retinopathy is a serious sight- hanging complication of diabetes. Diabetes interferes with the body's capability to use and store sugar(glucose).

The complaint is characterized by too important sugar in the blood, which can beget damage throughout the body, including the eyes. Over time, diabetes damages small blood vessels throughout the body, including the retina. Diabetic retinopathy occurs when these bitsy blood vessels blunder blood and other fluids. This causes the retinal towel to swell, performing in cloudy or blurred vision. The condition generally affects both eyes. The longer a person has diabetes, the more likely they will develop diabetic retinopathy. However, diabetic retinopathy can beget blindness, If left undressed.

Symptoms of diabetic retinopathy include

- Seeing spots or floaters
- Blurred vision
- Having a dark or empty spot in the center of your vision
- Difficulty seeing well at night

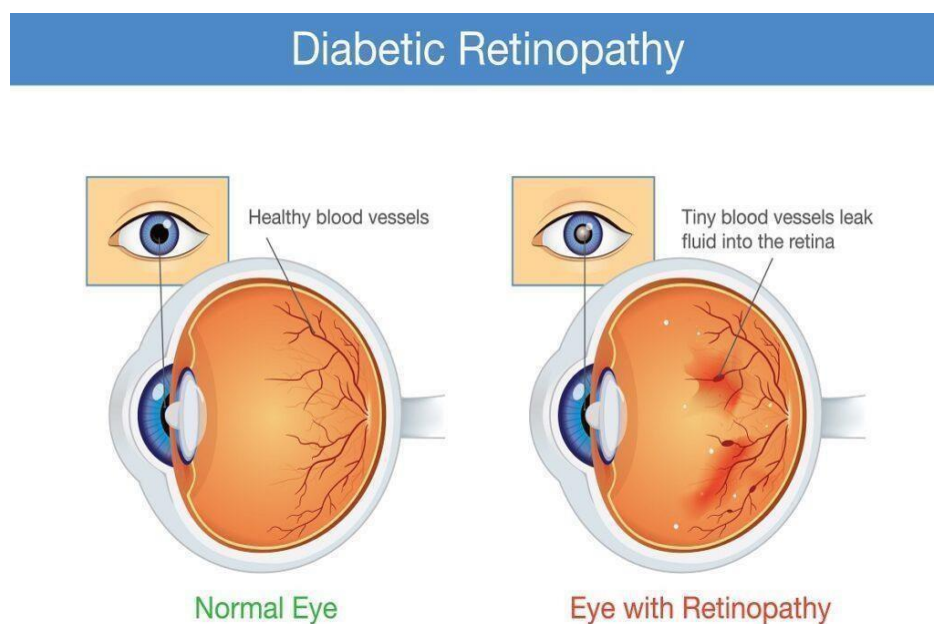
When people with diabetes experience long ages of high blood sugar, fluid can accumulate in the lens inside the eye that controls fastening. This changes the curve of the lens, leading to changes in vision. still, formerly blood sugar situations are controlled, generally the lens will return to its original shape and vision improves. Cases with diabetes who can more control their blood sugar situations will decelerate the onset and progression of diabetic retinopathy. According to a 2018 American Eye Q Survey conducted by the AOA, nearly half of Americans did not know whether diabetic eye conditions have visible symptoms (frequently which the early stages of diabetic retinopathy does not). The same check set up that further than one- third of Americans did not know a comprehensive eye test is the only way to determine if a person's diabetes will beget blindness, which is why the AOA recommends that everyone with diabetes have a comprehensive ballooned eye examination at least once a time. Beforehand discovery and treatment can limit the eventuality for significant

vision loss from diabetic retinopathy. Treatment of diabetic retinopathy varies depending on the extent of the complaint. People with diabetic retinopathy may need ray surgery to seal oohing blood vessels

or to discourage other blood vessels from oohing. Your Croaker of Optometry might need to fit specifics into the eye to drop inflammation or stop the conformation of new blood vessels. People with advanced cases of diabetic retinopathy might need a surgical procedure to remove and replace the gel-suchlike fluid in the reverse of the eye, called the vitreous. Surgery may also be demanded to repair a retinal detachment. This is a separation of the light-entering filling in the reverse of the eye. However, you can help or decelerate the development of diabetic retinopathy by

If you're diabetic. • Taking your specified drug

- Sticking to your diet
- Exercising regularly
- Controlling high blood pressure
- Avoiding alcohol and smoking



# **LITERATURE SURVEY**

## **Existing System:**

One of the conditions that poses the topmost threat to vision is diabetic retinopathy (DR), a consequence brought on by elevated blood glucose situations. Unfortunately, an ophthalmologist must manually collect DR webbing, which is time- consuming and subject to error. The enormous rise in the number of diabetic cases has led to an emphasis on automated DR opinion in recent times.

## **DISADVANTAGES OF Being SYSTEM:**

- must manually collect DR screening
- which is time- consuming
- which is subject to error

## **Proposed System:**

In this design, we probe the efficacy of light- weight deep literacy armature for fast and robust inflexibility grading of diabetic retinopathy. Our frame is grounded on a modified interpretation of vgg16 with integrating an attention medium with the former armature for further point refinement. likewise, we observe the effect of data imbalance on the model performance and alleviate such an effect by using an imbalanced literacy fashion.

## **ADVANTAGES OF PROPOSED SYSTEM:**

- Our network showed high faculty for inflexibility grading.
- The significant donation of the proposed frame is that it efficiently grades the inflexibility position of diabetic retinopathy while reducing the time and space complexity needed, which demonstrates it as a promising seeker for independent opinion

## **What causes diabetic retinopathy?**

Diabetic retinopathy results from the damage diabetes causes to the small blood vessels located in the retina. These damaged blood vessels can beget vision loss

- Fluid can blunder into the macula, the area of the retina responsible for clear central vision. Although small, the macula is the part of the retina that allows us to see colors and fine detail. The fluid causes the macula to swell, performing in blurred vision.
- In an attempt to ameliorate blood rotation in the retina, new blood vessels may form on its face. These fragile, abnormal blood vessels can blunder blood into the reverse of the eye and block vision.

**Non-proliferative diabetic retinopathy (NPDR)** is the early stage of the complaint in which symptoms will be mild or absent. In NPDR, the blood vessels in the retina are weakened. bitsy bulges in the blood vessels, called microaneurysms, may blunder fluid into the retina. This leakage may lead to swelling of the macula.

**Proliferative diabetic retinopathy (PDR)** is the more advanced form of the complaint. At this stage, rotation problems deprive the retina of oxygen. As a result, new, fragile blood vessels can begin to grow in the retina and into the vitreous, the gel- suchlike fluid that fills the reverse of the eye. The new blood vessels may blunder blood into the vitreous, clouding vision.

### **How is diabetic retinopathy diagnosed?**

Diabetic retinopathy can be diagnosed through a comprehensive eye examination. Testing with emphasis on assessing the retina and macula may include

- Case history to determine vision difficulties, presence of diabetes, and other general health enterprises that may be affecting vision
- Visual perceptivity measures to determine how important central vision has been affected
- Refraction to determine if a new eyeglass tradition is demanded
- Evaluation of the optical structures, including the evaluation of the retina through a ballooned pupil dimension of the pressure within the eye

## Convolutional Neural Network:

CNN is an artificial neural network armature that aims at learning low and high position features of medical images in an automated manner which helps in the discovery, bracket, and staging of medical conditions. CNN generally consists of several layers including

- convolution subcaste
- Pooling subcaste
- Completely connected (FC) subcaste

The convolutional subcaste is the first subcaste of a convolutional network. While convolutional layers can be followed by fresh convolutional layers or pooling layers, the completely- connected subcaste is the final subcaste. The affair of each subcaste is called an activation or a point chart, which can be an input to another subcaste. With each subcaste, the CNN increases in its complexity, relating lesser portions of the image. before layers concentrate on simple features, similar as colors and edges. As the image data progresses through the layers of the CNN, it starts to fete larger rudiments or shapes of the object until it eventually identifies the willed object.

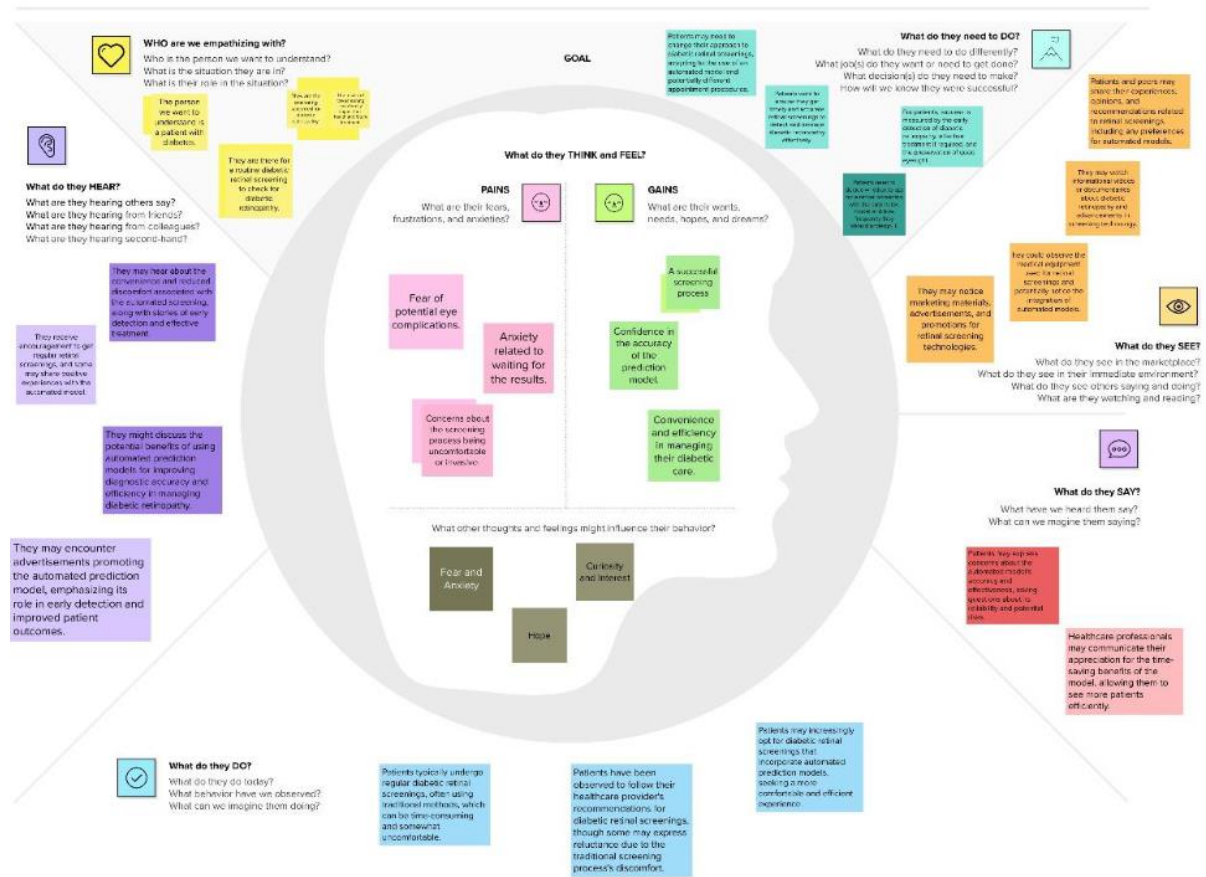
- Convolutional subcaste A set of direct pollutants is applied to the input image or the activation chart in the convolutional subcaste to prize a number of different low or high position features similar as edges, angles, blood vessels, etc. The affair of a  $3 \times 3$  complication is defined as follows

$$y(l, m, n) = \sum_{k=1}^3 \sum_{i=1}^3 \sum_{j=1}^3 w(l, i, j, k) x(i + m - 1, j + n - 1, k) + b(l)$$

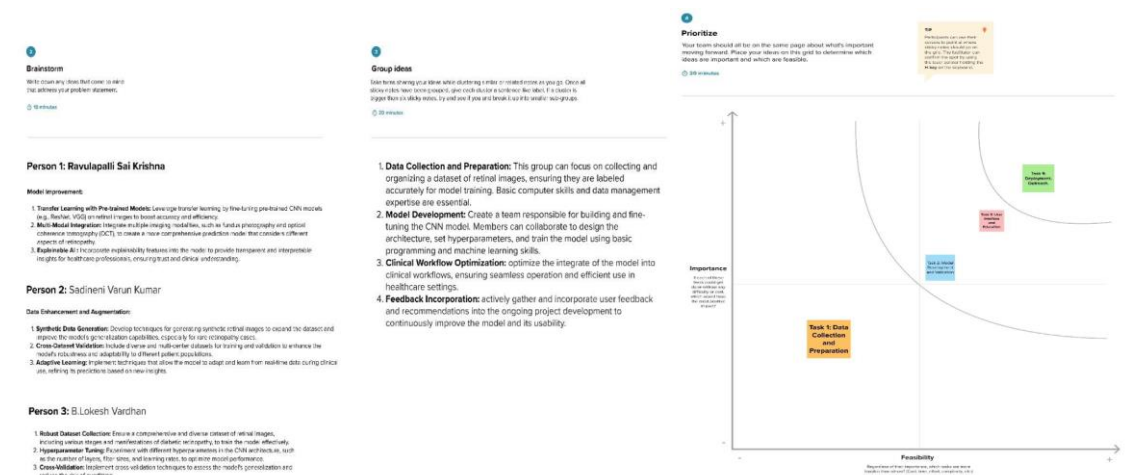
where  $x(i, j, k)$  the image argentine position value, and  $w(l, i, j, k)$  and  $b(l)$  represent the weights and impulses, independently, of the convolutional subcaste.

# IDEATION & PROPOSED SOLUTION

## Empathy Map Canvas:



## Ideation & Brainstorming:





## **REQUIREMENT ANALYSIS**

To be used efficiently, all computer software needs certain tackle factors or other software coffer to be present on a computer. These prerequisites are known as (computer) system conditions and are frequently used as a guideline as opposed to an absolute rule. utmost software defines two sets of system conditions minimal and recommended. With adding demand for advanced processing power and coffer in newer performances of software, system conditions tend to increase over time. Assiduity judges suggest that this trend plays a bigger part in driving upgrades to being computer systems than technological advancements.

### **NON-FUNCTIONAL Conditions:**

Inoperative conditions are the functions offered by the system. It includes time constraints and constraints on the development process and norms. The non-functional conditions are as follows  
Speed The system should reuse the given input into affair within applicable time.

Ease of use the software should be stoner friendly. Also the guests can use fluently, so it doesn't bear important training time.

Trustability The rate of failures should be lower also only the system is more dependable  
Portability It should be easy to apply in any system.

### **SPECIFIC Conditions**

The specific conditions are:

stoner Interfaces The external druggies are the guests. All the guests can use this software for indexing and searching.

Hardware Interfaces The external tackle interface used for indexing and searching is particular computers of the guests. The PC's may be laptops with wireless LAN as the internet connections handed will be wireless.

Software Interfaces The Operating Systems can be any interpretation of Windows.

Performance Conditions The PC's used must be at least Pentium 4 machines so that they can give optimum performance of the product.

### **SOFTWARE SPECIFICATIONS:**

Software conditions deal with defining software resource conditions and prerequisites that need to be installed on a computer to give optimal functioning of an operation.

These conditions or prerequisites are generally not included in the software installation package and need to be installed independently before the software is installed.

The system should be suitable to affiliate with the being system

- The system should be accurate
- The system should be better than the being system tackle SPECIFICATIONS

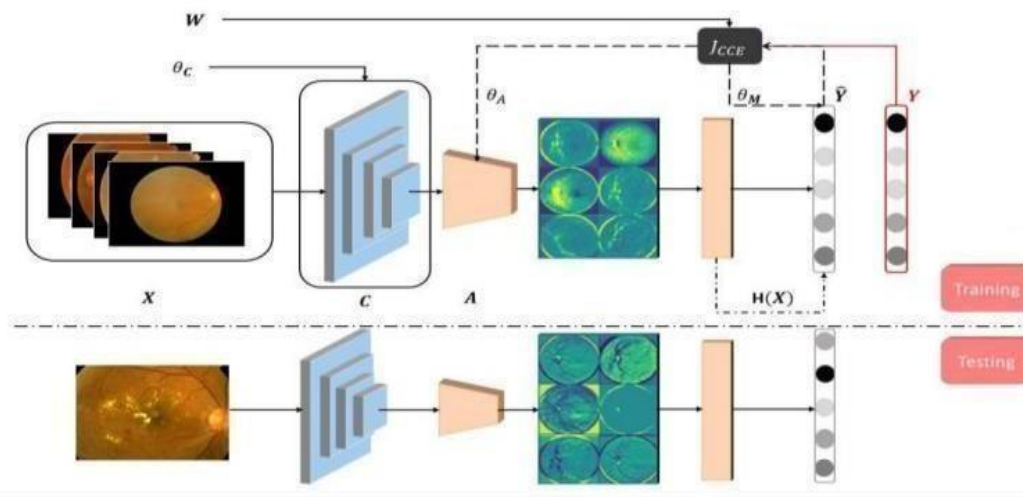
The most common set of conditions defined by any operating system or software operation is the physical computer coffers, also known as tackle, A tackle conditions list is frequently accompanied by a tackle comity list, especially in case of operating systems. An HCL lists tested, compatible, and occasionally inharmonious tackle bias for a particular operating system or operation. The following sub-sections bandy the colorful aspects of tackle conditions.

All computer operating systems are designed for a particular computer armature. utmost software operations are limited to particular operating systems running on particular infrastructures. Although armature-independent operating systems and operations live, utmost need to be reedited to run on a new armature.

The power of the central processing unit (CPU) is a abecedarian system demand for any software. utmost software running on x86 armature define processing power as the model and the timepiece speed of the CPU. numerous other features of a CPU that impact its speed and power, like machine speed, cache, and MIPS are frequently ignored.

## PROJECT DESIGN

Systems design is the process of defining the armature, factors, modules, interfaces, and data for a system to satisfy specified conditions. It may be considered the operation of systems proposition to the process of product development. The most popular ways for designing computer systems are decreasingly those that concentrate on object- acquainted analysis and design.



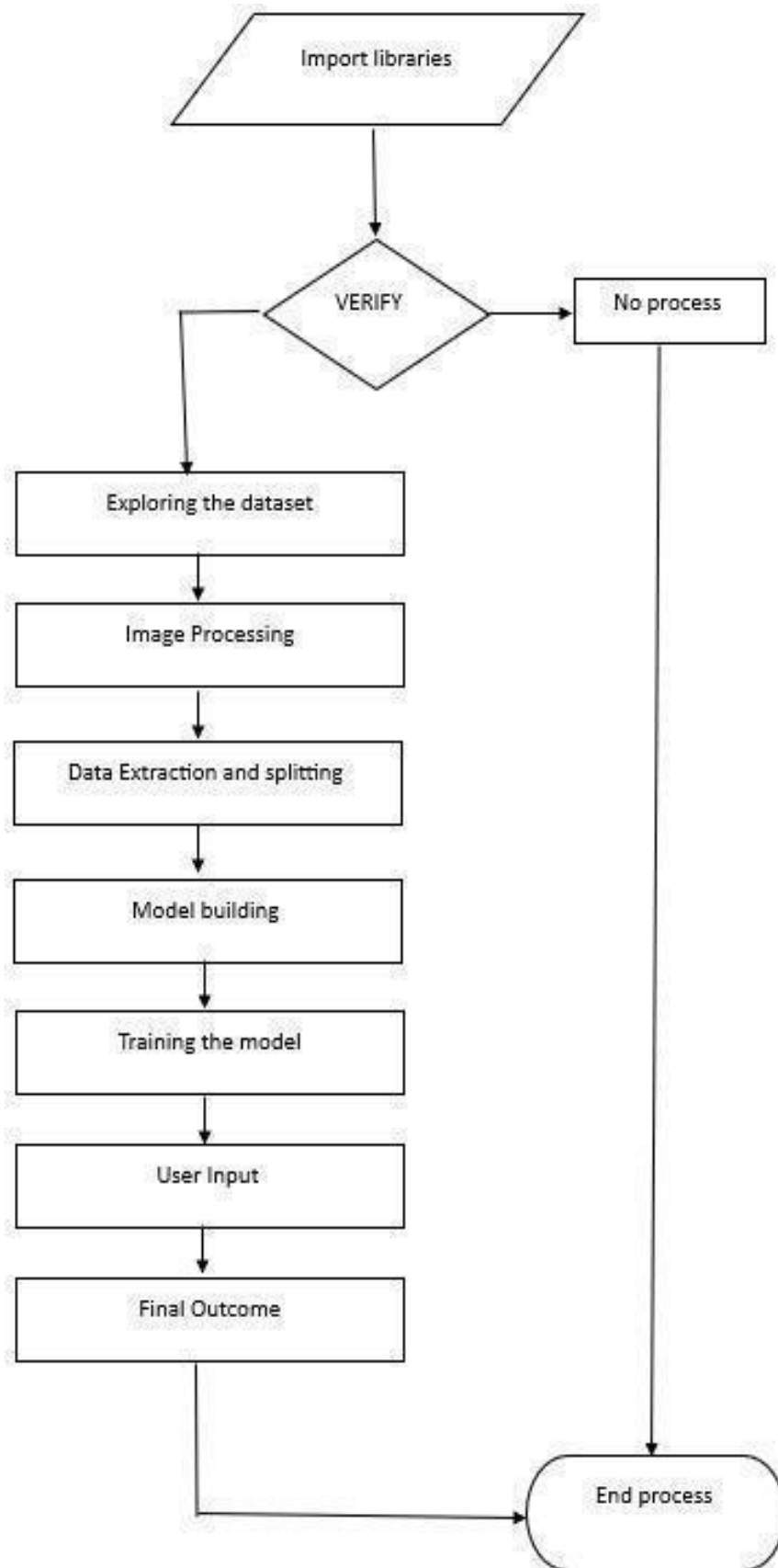
The above fig shows CNN model architecture for Diabetic Retinopathy

### Data Flow Diagram

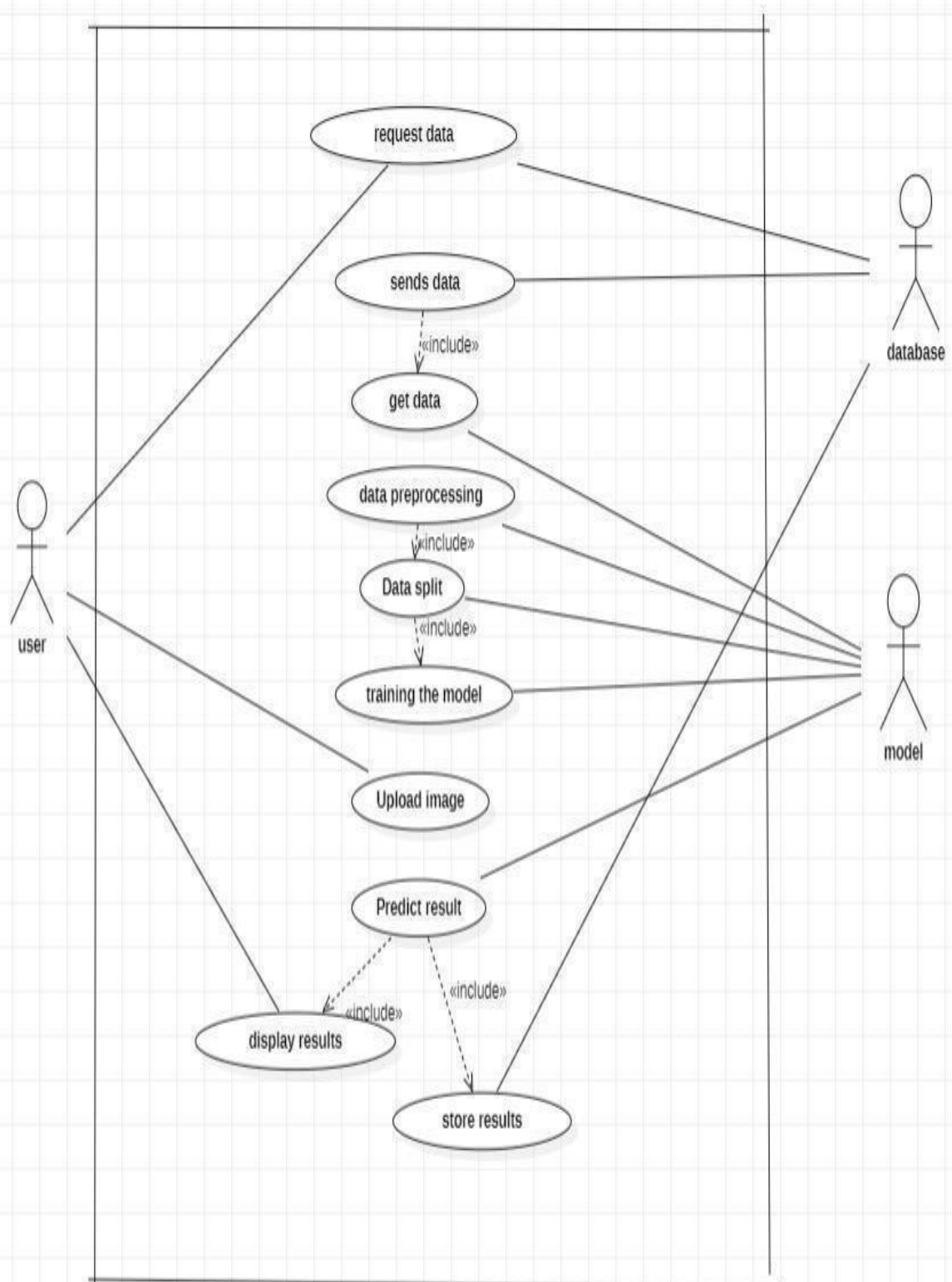
A Data Flow Diagram(DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right quantum of the system demand graphically. It can be homemade, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

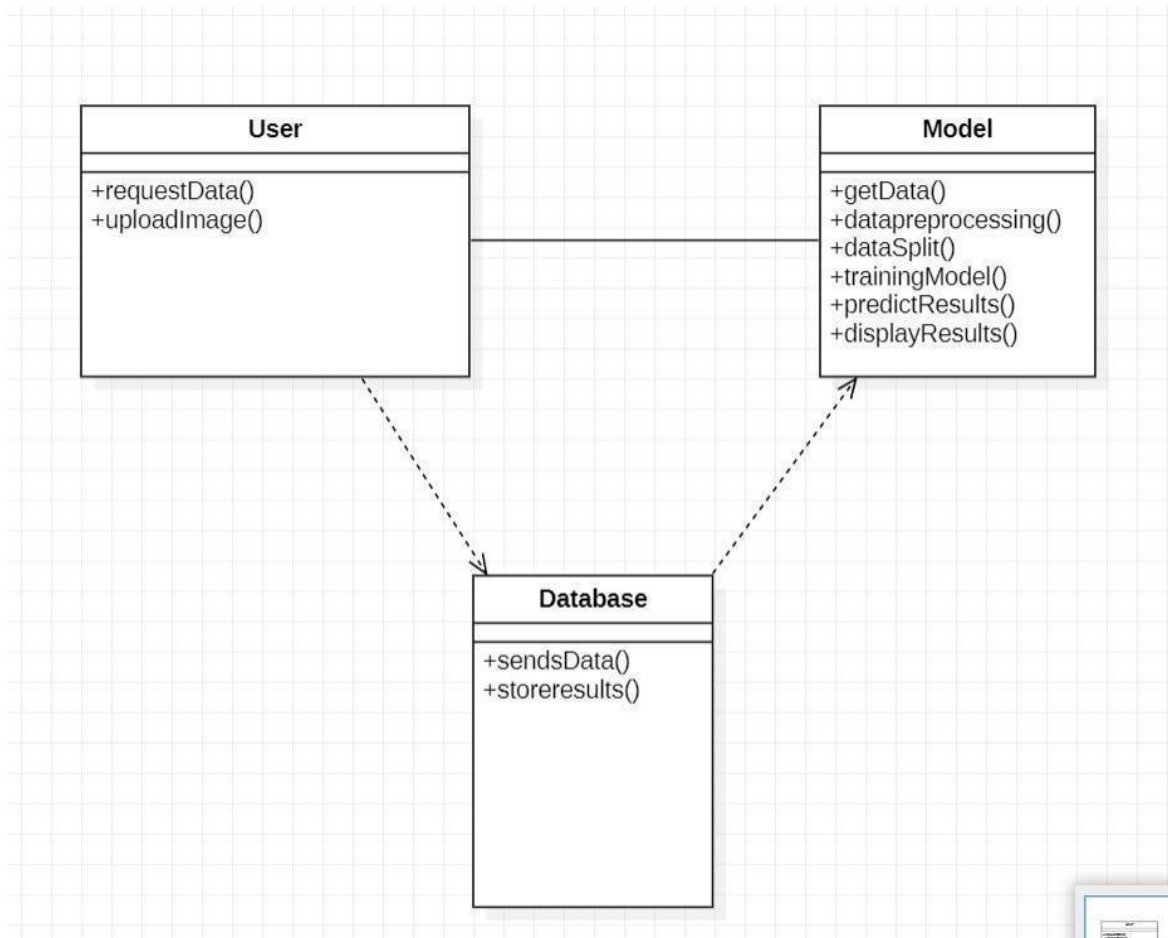
The ideal of a DFD is to show the compass and boundaries of a system as a whole. It may be used as a communication tool between a system critic and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data inflow graph or bubble map.



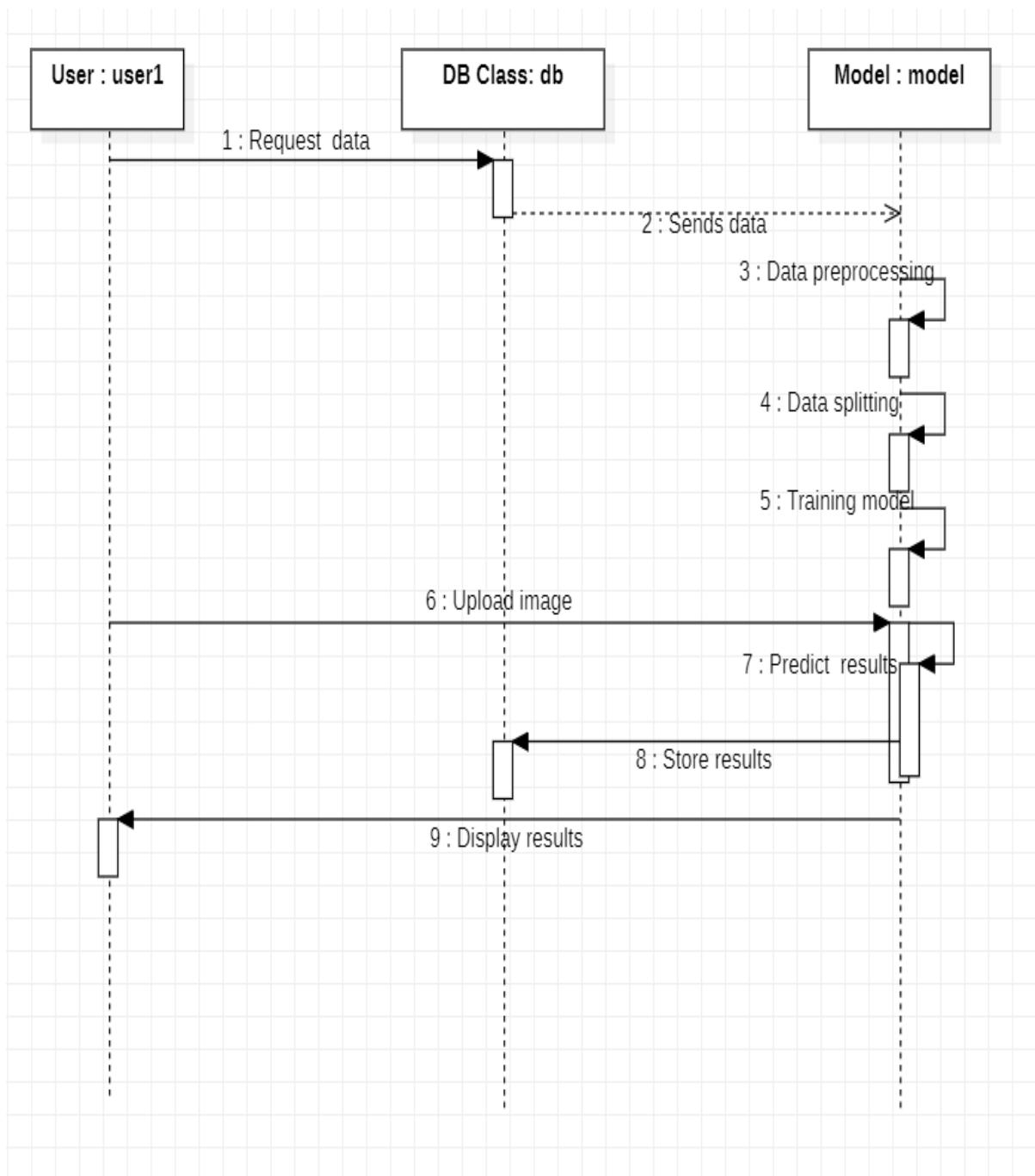
**Data flow Diagram**



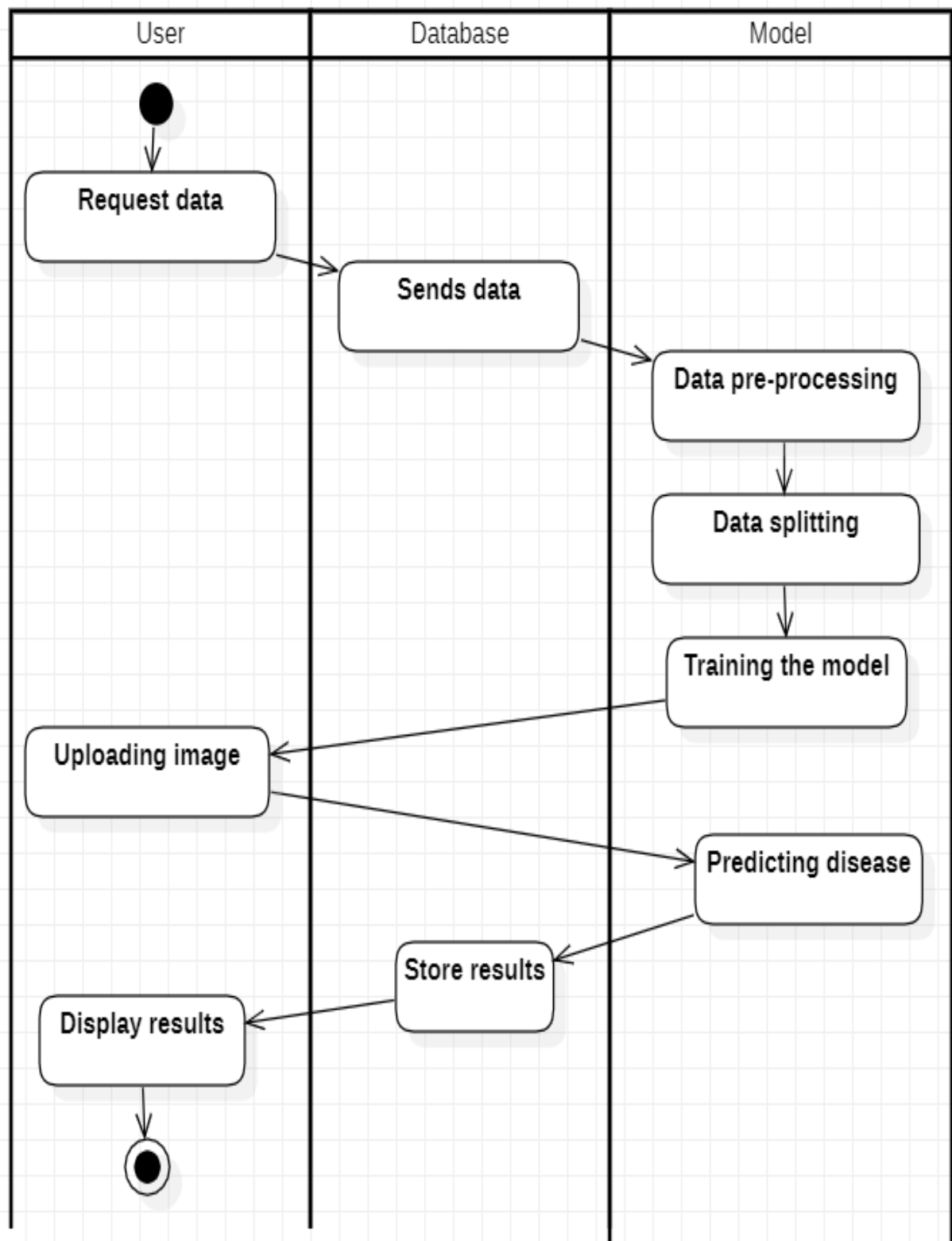
**Use Case Diagram**



**Class Diagram**



**Sequence Diagram**



**Activity Diagram**



## User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Healthcare Professional	Diabetic Retinopathy Prediction	US01	Upload Retinal Images	The user can upload retinal images to the system.	High	Sprint 1
Diabetic Patient	Diabetic Retinopathy Prediction	US02	View DR Predictions	The patient can view their DR predictions.	High	Sprint 1
Data Scientist	Data Access & Preparation	US03	Access Dataset	The data scientist can access the dataset.	Medium	Sprint 1
Hospital Administrator	Efficiency & Automation	US06	System Cost-Effectiveness	The system reduces healthcare staff workload.	High	Sprint 1

## **Solution Architecture:**

The solution architecture of a Diabetic Retinopathy classification model typically involves several components and steps. Here's a high-level overview of the architecture for building and deploying such a model:

### **Data Collection and Preprocessing:**

**Data Collection:** Gather a dataset of retinal images, where each image is labeled with the corresponding Diabetic Retinopathy severity level.

**Data Preprocessing:** Prepare the data by resizing images to a consistent size, normalizing pixel values, and augmenting the dataset if necessary. Ensure that the data is split into training, validation, and test sets.

### **Model Selection:**

Choose an appropriate machine learning or deep learning model for image classification. Convolutional Neural Networks (CNNs) are commonly used for this task due to their ability to capture image features effectively.

### **Model Training:**

Train the selected model on the training dataset. Use techniques like transfer learning if available pre-trained models such as VGG16, ResNet, or Inception to leverage their feature extraction capabilities.

### **Hyperparameter Tuning:**

Optimize hyperparameters, such as learning rate, batch size, and the number of layers in your model, to improve performance. Use techniques like grid search or random search to find the best hyperparameters.

**Model Evaluation:** Evaluate the model's performance using the validation dataset. Common evaluation metrics include accuracy, precision, recall, F1-score, and the confusion matrix.

### **Fine-Tuning:**

If the model's performance is not satisfactory, fine-tune the model by adjusting hyperparameters, changing the architecture, or increasing the size of the training dataset.

### **Deployment:**

Once the model performs well on the validation dataset, deploy it to a production environment where it can make predictions on new, unseen data.

Create an API or web service for integration with other systems or applications.

Implement monitoring and logging to track model performance over time.

### **User Interface:**

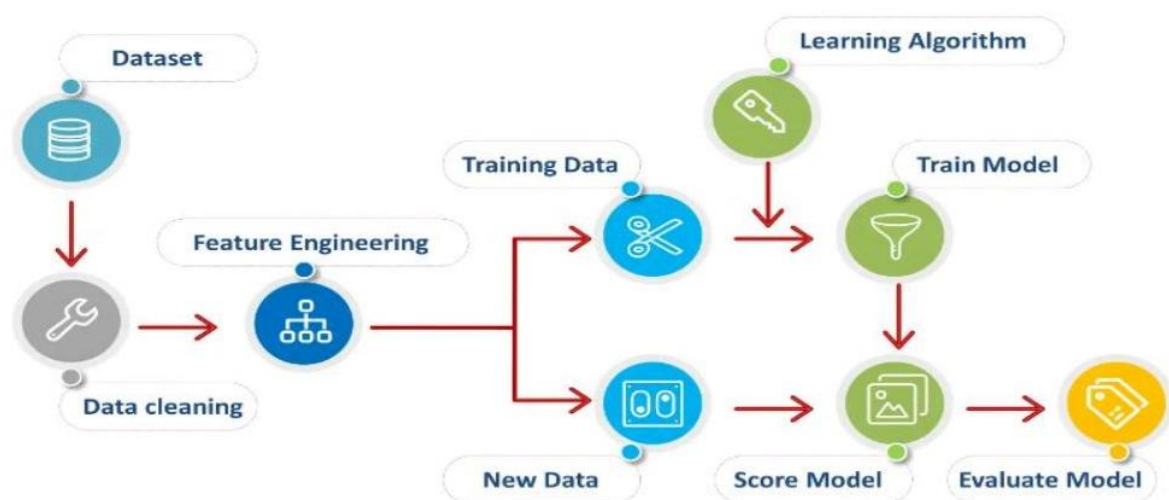
Develop a user interface that allows users to upload retinal images and receive predictions regarding Diabetic Retinopathy severity.

### **Documentation:**

Create comprehensive documentation for the entire solution, including model architecture, deployment procedures, and user instructions.

### **Testing and Quality Assurance:**

Thoroughly test the entire solution, including the model, user interface, and deployment, to ensure it functions as expected.



# PROJECT PLANNING & SCHEDULING

## Technical Architecture

Component	Description	Technologies
User Interaction	Interface for user interaction with the application along with creating an user friendly interface	Web-based UI (HTML, CSS, JavaScript) or StreamLit platform
Application Logic-1	Core logic responsible for handling user requests	Python, Flask,
Data Collection	Data collection encompasses sourcing historical Bitcoin price data from exchanges via APIs, crucial for Time Series Analysis using Prophet	Integrate Python, Prophet, and cloud infrastructure for accurate predictions.
Data Input	Handling and processing user-provided input data	Forms, file uploads, APIs, or command-line input.
External API	Integration with external data sources or services.	RESTFUL APIs(Bitcoin API, Binance API)
Cloud Database	Storage and management of structured data	Amazon RDS, Google Cloud SQL, or Azure SQL.
Model Integration	Interface for integrating with FbProphet Model	RESTful API endpoints (Flask, FastAPI,JSON,XML
API Model Deployment	Responsible for deploying machine learning models as APIs, enabling real-time predictions and external interaction. It ensures model accessibility and scalability	Docker, Kubernetes, or serverless (AWS Lambda).
Machine Learning Model	The predictive model using Fb Prophet for bitcoin forecasting	Scikit-Learn, Keras TensorFlow, PyTorch, XGBoost, fb Prophet

Data Preprocessing	Data preparation and feature engineering.	Pandas, NumPy, scikit-learn, or custom scripts.
Model Deployment	Hosting and serving the machine learning model.	Flask, FastAPI, TensorFlow Serving, or StreamLit.
Infrastructure (Server/Cloud)	Underlying cloud infrastructure and resources	AWS, Google Cloud, Azure, or on-premises servers like Local, Cloud Foundry, Kubernetes etc.

**Table 2: Application Characteristics:**

Component	Description	Technologies
Open-Source Frameworks	Leveraging open-source frameworks for model development and deployment guarantees cost-efficiency and flexibility, streamlining user access to Bitcoin price predictions, especially during market fluctuations and high volatility periods, enhancing accuracy and reliability in the Time Series Analysis project for Bitcoin Price Prediction using Prophet.	- Scikit-Learn, TensorFlow, PyTorch for model development. - Flask or FastAPI for API deployment. - Kubernetes for container orchestration. - Jupyter Notebook for model prototyping and development
Security Implementations	Implementations include robust encryption, authentication, and access controls to safeguard sensitive data in the Time Series Analysis project for Bitcoin Price Prediction using Prophet, leveraging Python, Prophet, and cloud-	- OAuth 2.0 or JWT for user authentication. - Encryption (HTTPS/SSL) for data in transit. - Role-based access control. - Regular security audits and updates. - Compliance with industry standards (e.g., GDPR)

	based security protocols for advanced protection.	
Scalable Architecture	Designing a scalable architecture that can handle growing data volumes and user demands which can manage the huge inflow of user demands assuming as a big data	<ul style="list-style-type: none"> <li>- Microservices architecture for modularity and scalability.</li> <li>- Containerization with Docker and orchestration with Kubernetes.</li> <li>- Load balancers for distributing traffic.</li> <li>- Auto-scaling based on resource usage.</li> </ul>
Availability	Ensuring high availability and minimal downtime for the Time Series Analysis application is vital, enabling continuous data processing and accurate Bitcoin price predictions using Prophet for optimal financial insights.	<ul style="list-style-type: none"> <li>Redundancy in database and API deployment.</li> <li>- Geographically distributed data centers or cloud regions.</li> <li>- Monitoring and alerting systems (e.g., Prometheus, Grafana).</li> <li>- Failover mechanisms for fault tolerance.</li> </ul>
Performance	Optimizing application performance to provide quick insights and predictions	<ul style="list-style-type: none"> <li>Caching mechanisms for frequently accessed data.</li> <li>- Model optimization (e.g., quantization) for faster inference.</li> <li>- Load testing and performance tuning</li> </ul>
User-Friendly Interface	Creating an intuitive and user friendly interface for data input, visualization, and interact	<ul style="list-style-type: none"> <li>HTML, CSS, JavaScript for web-based UI.</li> <li>- React or similar frameworks for responsive design.</li> <li>- Data visualization libraries (e.g., D3.js).</li> <li>- User experience (UX) testing and design principles.</li> </ul>

## Sprint Planning & Delivery Schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1,2	Data Collection and Preparation	USN-1	As a Data Scientist, I want to access reliable historical Bitcoin price data, so I can analyze and predict trends effectively.	1	High	Varun Kumar, Sai Krishna
Sprint-3,4	Model Development and Training	USN-2	As a Machine Learning Engineer, I want to create a baseline Prophet model for Bitcoin price prediction, so I can establish a foundation for further optimization.	2	High	Sai Krishna, Lokesh
Sprint-5,6	Evaluation and Validation	USN-3	As a Product Owner, I want to visualize model predictions against actual Bitcoin prices, so I can interpret the results and make informed decisions.	3	High	Varun, Sai Krishna, Lokesh
Sprint-7,8	Front-End Development	USN-4	As a Front-End Developer, I want to implement interactive charts on the user interface, so	5	High	Varun, Lokesh

			users can visualize both predicted and actual Bitcoin prices dynamically.			
Sprint-9,10	Cloud Infrastructure and Deployment	USN-5	As a DevOps Engineer, I want to select an appropriate cloud service provider and set up infrastructure, so the application and model can be hosted securely and efficiently.	6	High	Lokesh
Sprint-11,12	Performance Optimization	USN-6	As a QA Engineer, I want to define comprehensive test objectives and scope, so I can ensure the application functions as intended and meets quality standards.	3	Medium	Varun
Sprint-13	Evaluation & Testing	USN-7	As a quality assurance specialist, I want to test the model's performance rigorously and evaluate its effectiveness for Bitcoin price prediction.	2	High	Sai Krishna

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1,2	8	1 Day	28 Oct 2023	28 Oct 2023	9	28 Oct 2023
Sprint-3,4	8	1 Day	29 Oct 2023	29 Oct 2023	8	29 Oct 2023
Sprint-5,6	8	3 Days	30 Oct 2023	01 Nov 2023	10	01 Nov 2023
Sprint-7,8,9	12	3 Days	02 Nov 2023	04 Nov 2023	11	04 Nov 2023

Sprint-10,11,12	12	2 Days	05 Nov 2023	06 Nov 2023	5	06 Nov 2023
-----------------	----	--------	-------------	-------------	---	-------------

## **CODING & SOLUTIONING**

The perpetration stage of any design is a true display of the defining moments that make a design a success or a failure. The installation and operationalization of the system or system variations in a product terrain is appertained to as the perpetration step. After the system has been tried out and approved by the stoner, the phase is started. This phase continues until the system is operating in product in agreement with the defined stoner conditions.

### **Language / Technology Used**

Python is the language used for developing the detection and process of the system and for image processing using Convolutional Neural Network.

### **Libraries / Algorithms Used**

#### **NUMPY**

NumPy is a python library used for working with arrays. It also has functions for working in the sphere of direct algebra, fourier transfigure, and matrices. NumPy was created in 2005 by Travis Oliphant. It's an open source design and you can use it freely. NumPy stands for Numerical Python. In Python we've lists that serve the purpose of arrays, but they're slow to reuse. NumPy aims to give an array object that's over to 50x faster than traditional Python lists. At the core of the NumPy package, is the array object. This encapsulates n- dimensional arrays of homogeneous data types, with numerous operations being performed in collected law for performance. There are several important differences between NumPy arrays and the standard Python sequences

#### **SCIKIT LEARN**

Scikit- learn(Sklearn) is the most useful and robust library for machine literacy in Python. It provides a selection of effective tools for machine literacy and statistical modeling including bracket, retrogression, clustering and dimensionality reduction via a harmonious interface in Python. This library, which is largely written in Python, is erected upon NumPy, SciPy and Matplotlib.



It was firstly called scikits learn and was originally developed by David Cournapeau as a Google summer of law design in 2007. latterly, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA( French Institute for Research in Computer Science and robotization), took this design at another position and made the first public release( v0.1 beta) on 1stFeb. 2010.

## **TENSORFLOW**

TensorFlow is a Python library for fast numerical computing. It was created and is maintained by Google and released under the Apache2.0 open source license. It's a foundation library that can be used to produce Deep literacy models directly or by using wrapper libraries that simplify the process erected on top of Tensor Flow.

calculation in TensorFlow is described in terms of data inflow and operations in the structure of a directed graph.

- **Bumps** perform calculation and have zero or further inputs and labors. Data that moves between bumps are known as tensors, which are multi-dimensional arrays of real values.
- **Edges** The graph defines the inflow of data, branching, looping and updates to state. Special edges can be used to attend geste within the graph, for illustration staying for calculation on a number of inputs to complete.
- **Operation** An operation is a named abstract calculation which can take input attributes and produce affair attributes. For illustration, you could define an add or multiply operation.

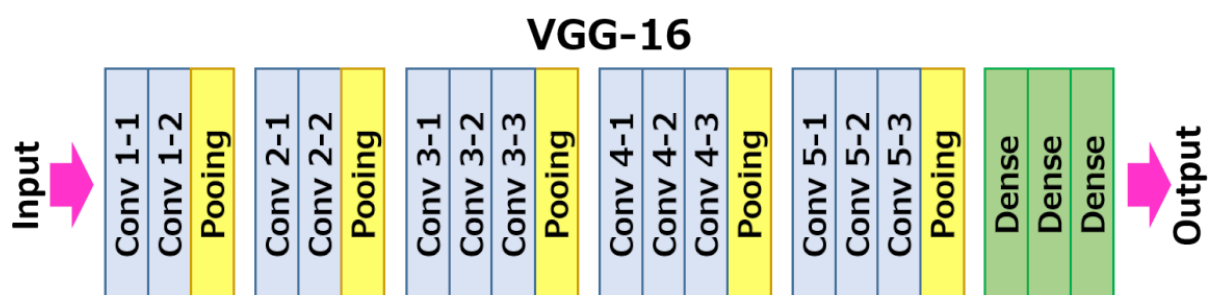
## **KERAS**

Keras is one of the most popular Python libraries for Deep literacy. Keras is simple, flexible and important. It's a library that provides you colorful tools to deal with neural network models. These tools enable you to fantasize and understand the model. These represent the factual neural network model. These models group layers into objects. There are two types of Models available in Keras The successional model and the Functional model.

Keras Sequential Model is simple and easy to use model. It's a direct mound of styles that groups a direct mound of layers into `atf.keras.Model`. According to its name, its main task is to arrange the layers of the Keras in successional order. In this model, the data inflow from one subcaste to another subcaste. The inflow of data is continued until the data reaches the final subcaste. utmost of the Neural Networks use the successional API Model

## VGG16

VGG16 is a simple and extensively used Convolutional Neural Network( CNN) Architecture. The VGG16 Architecture was developed and introduced by Karen Simonyan and Andrew Zisserman from the University of Oxford. VGG16 is used in numerous deep literacy image bracket ways and is popular due to its ease of perpetration. It was one of the notorious model with large kernel- sized pollutants( 11 and 5 in the first and alternate convolutional subcaste, independently) with multiple  $3 \times 3$  kernel- sized pollutants one after another.



### Configurations:

The ConvNet configurations are outlined in figure \*\*. The nets are appertained to their names(A-E). All configurations follow the general design present in armature and differ only in the depth from 11 weight layers in the network A(8 conv. and 3 FC layers) to 19 weight layers in the network E( 16 conv. and 3 FC layers). The range of conv. layers the number of channels) is rather small, starting from 64 in the first subcaste and also adding by a factor of 2 after each maximum-pooling subcaste, until it reaches 512.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

## Sample Code:

## IMPORTING DEPENDENCIES:

```
[2] from google.colab import drive
drive.mount('/content/drive/')

Mounted at /content/drive/

[3] !pip install -q keras

[4] !pip install tensorflow

Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: absl-py>=0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast>=0.5.0, <0.5.1, in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf>=4.21.0, <4.21.1, <4.21.2, <4.21.3, <4.21.4, <4.21.5, <5.0.0dev, >=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=4.6.4 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt>=1.15, <1.16.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)
Requirement already satisfied: grpcio>=2.9, <2.34.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.2)
Requirement already satisfied: tensorflow-estimator>=2.15, <2.16.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.1)
Requirement already satisfied: tensorflow-estimator>=2.15, <2.16.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: keras>=2.15, <2.16.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.16.0)
Requirement already satisfied: wheel<1.0, >=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.41.3)
Requirement already satisfied: google-auth>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1, >=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.8 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (3.5.1)
Requirement already satisfied: requests>2.9, <2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (2.31.0)
Requirement already satisfied: tensorflow-data-server>=0.8.0, <0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=0.8 in /usr/local/lib/python3.10/dist-packages (from tensorflow-estimator>=2.15->tensorflow) (0.8.4)
Requirement already satisfied: cachetools>=4.6, <5.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth>=1.6.3->tensorflow-estimator) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth>=1.6.3->tensorflow-estimator) (0.3.0)
Requirement already satisfied: rsa>=4.1, <4.9 in /usr/local/lib/python3.10/dist-packages (from google-auth>=1.6.3->tensorflow-estimator) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1, >=0.5->tensorflow-estimator) (1.3.1)
Requirement already satisfied: charset-normalizer>=2.8 in /usr/local/lib/python3.10/dist-packages (from requests>2.9->tensorflow-estimator) (3.3.2)
Requirement already satisfied: idna>=3, <2.5 in /usr/local/lib/python3.10/dist-packages (from requests>2.9->tensorflow-estimator) (3.4)
Requirement already satisfied: urllib3>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>2.9->tensorflow-estimator) (2.0.7)
Requirement already satisfied: certifi>=2017.4.7 in /usr/local/lib/python3.10/dist-packages (from requests>2.9->tensorflow-estimator) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=0.8->tensorflow-estimator) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0, >=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth>=1.6.3->tensorflow-estimator) (0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1, >=0.5->tensorflow-estimator) (3.2.2)

[5] import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt
%matplotlib inline
```

## Data Collection and analysis:

Setting up the file paths of the training, and validation data. Creating batches of data using the ImageDataGenerator

```
[6] test_path = '/content/drive/MyDrive/Colab Notebooks/ml_project/test'
train_path = '/content/drive/MyDrive/Colab Notebooks/ml_project/train'
valid_path = '/content/drive/MyDrive/Colab Notebooks/ml_project/valid'

[7] from tensorflow.keras.preprocessing.image import ImageDataGenerator

test_batches = ImageDataGenerator().flow_from_directory(test_path, target_size=(224, 224), classes=['dr', 'nodr'], batch_size=10)
train_batches = ImageDataGenerator().flow_from_directory(train_path, target_size=(224, 224), classes=['dr', 'nodr'], batch_size=10)
valid_batches = ImageDataGenerator().flow_from_directory(valid_path, target_size=(224, 224), classes=['dr', 'nodr'], batch_size=10)

Found 40 images belonging to 2 classes.
Found 40 images belonging to 2 classes.
Found 40 images belonging to 2 classes.
```

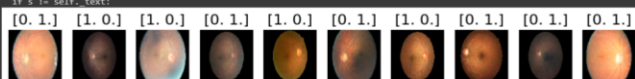
Defining a function named plots for plotting images with their corresponding label

```
[8] def plots(ims, figsize=(12,6), rows=1, interp=False, titles=None):
    if type(ims[0]) is np.ndarray:
        ims = np.array(ims).astype(np.uint8)
        if(ims.shape[-1] != 3):
            ims = ims.transpose((0,2,3,1))
    f = plt.figure(figsize=figsize)
    cols = len(ims)//rows if len(ims) % 2 == 0 else len(ims)//rows + 1
    for i in range(len(ims)):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('off')
        if titles is not None:
            sp.set_title(titles[i], fontsize=10)
        plt.imshow(ims[i], interpolation=None if interp else 'none')

[9] imgs, labels = next(train_batches)

[10] plots(imgs, titles=labels)

/usr/local/lib/python3.10/dist-packages/matplotlib/text.py:1279: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
if s >= self._text:
```



```

10a [11] from tensorflow import keras
      from tensorflow.keras.applications import VGG16

      # Create a VGG16 model
      vgg16_model = VGG16()

      Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels.h5
      553467096/553467096 [=====] - 6s 0us/step

```

## Loading the pre-Trained VGG16 model provided by keras

```

[12] vgg16_model = keras.applications.vgg16.VGG16()

vgg16_model.summary()

Model: "vgg16"

```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

```

Total params: 148357984 (527.79 MB)
Trainable params: 138357984 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)

```

```

[14] type(vgg16_model)

keras.src.engine.functional.Functional

[15] from tensorflow.keras.models import Sequential # Import the Sequential class

model = Sequential() # Create a Sequential model

for layer in vgg16_model.layers[:-1]:
    model.add(layer) # Add layers to the model

model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312

```

Total params: 134260544 (512.16 MB)
Trainable params: 134260544 (512.16 MB)
Non-trainable params: 0 (0.00 Byte)

```

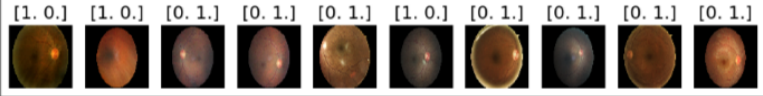
## Compiling the model

Fitting the model on the training data and validating on the validation data.

```
[21] model.fit(train_batches, steps_per_epoch=4,
               validation_data=valid_batches, validation_steps=4, epochs=5, verbose=2)

Epoch 1/5
4/4 - 79s - loss: 0.9520 - accuracy: 0.4500 - val_loss: 0.8186 - val_accuracy: 0.4500 - 79s/epoch - 28s/step
Epoch 2/5
4/4 - 64s - loss: 0.8458 - accuracy: 0.3750 - val_loss: 0.8482 - val_accuracy: 0.4750 - 64s/epoch - 16s/step
Epoch 3/5
4/4 - 65s - loss: 0.8421 - accuracy: 0.4750 - val_loss: 0.8448 - val_accuracy: 0.4750 - 65s/epoch - 16s/step
Epoch 4/5
4/4 - 46s - loss: 0.7756 - accuracy: 0.4750 - val_loss: 0.8078 - val_accuracy: 0.4250 - 46s/epoch - 12s/step
Epoch 5/5
4/4 - 65s - loss: 0.6977 - accuracy: 0.6500 - val_loss: 0.8087 - val_accuracy: 0.4250 - 65s/epoch - 16s/step
<keras.src.callbacks.History at 0x7a29decdd17e0>

[22] test_imgs, test_labels = next(test_batches)
     plots(test_imgs, titles=test_labels)

[1. 0.] [1. 0.] [0. 1.] [0. 1.] [0. 1.] [1. 0.] [0. 1.] [0. 1.] [0. 1.] [0. 1.]


[23] test_labels = test_labels[:,0]
     test_labels

array([1., 1., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

import matplotlib.pyplot as plt
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Loading the testing data and making predictions using the predict method of the model. Plotting the confusion matrix of the model's predictions.

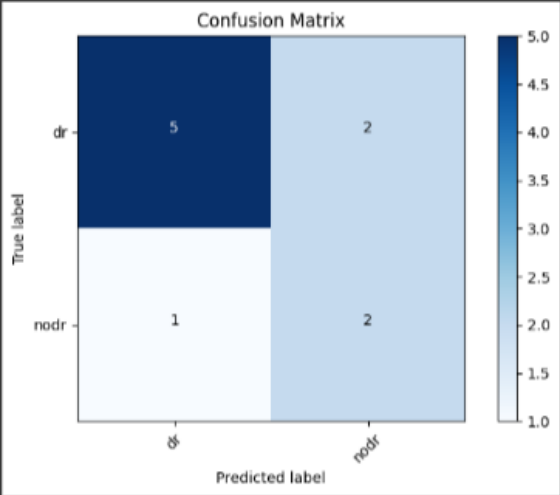
```
[26] from sklearn.metrics import confusion_matrix # Import the confusion_matrix function

cm = confusion_matrix(test_labels, np.round(predictions[:, 0]))
```

```
[27] import itertools # Import the itertools module

cm_plot_labels = ['dr', 'nodr']
plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
```

Confusion matrix, without normalization  
[[5 2]  
[1 2]]



```
from sklearn.metrics import accuracy_score

# Assuming you have the ground truth labels in test_labels and model predictions in predictions
accuracy = accuracy_score(test_labels, np.round(predictions[:, 0]))
print("Accuracy Score:", accuracy)
```

Accuracy Score: 0.7

```
[43] from sklearn.metrics import classification_report

# Assuming you have the ground truth labels in test_labels and model predictions in predictions
class_report = classification_report(test_labels, np.round(predictions[:, 0]), target_names=cm_plot_labels)
print("Classification Report:")
print(class_report)
```

	precision	recall	f1-score	support
dr	0.83	0.71	0.77	7
nodr	0.50	0.67	0.57	3
accuracy			0.70	10
macro avg	0.67	0.69	0.67	10
weighted avg	0.73	0.70	0.71	10

```
[28] model.save('diabetic_retinopathy.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3679: UserWarning: You are saving your model as an HDF5 file via 'model.save()'. This file format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my_model.keras')'.
  saving_api.save_model(
```

Defining a function named `fix_layer0` for fixing the batch input shape and data type of the model. Defining a function named `get_model` for loading the pre-trained model.

Defining a function named `preprocess_image` for preprocessing an image before making a prediction.

```
5s [▶] def fix_layer0(filename, batch_input_shape, dtype):
    with h5py.File(filename, 'r+') as f:
        model_config = json.loads(f.attrs['model_config'])
        layer0 = model_config['config']['layers'][0]['config']
        layer0['batch_input_shape'] = batch_input_shape
        layer0['dtype'] = dtype
        f.attrs['model_config'] = json.dumps(model_config).encode('utf-8')

def get_model():
    global model, graph
    model = tf.keras.models.load_model('diabetic_retinopathy.h5', compile=False)
    print(" * Model Loaded!!")
    graph = tf.get_default_graph()

def preprocess_image(image, target_size):
    if image.mode != "RGB":
        image = image.convert("RGB")
    image = image.resize(target_size)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    return image

print(" * Loading keras model.....")
fix_layer0('diabetic_retinopathy.h5', [None, 224, 224, 3], 'float32')
get_model()

[▶] * Loading keras model.....
    * Model Loaded!!
```



# RESULTS

```
10s import tensorflow as tf

# assuming `prediction` is a symbolic tensor
image = Image.open('/content/drive/MyDrive/Colab_Notebooks/ml_project/test/dr/54_right.jpeg')
image.show()
processed_image = preprocess_image(image, target_size=(224, 224))
prediction = model.predict_on_batch(processed_image)
prediction_shape = tf.shape(prediction)
num_samples = prediction_shape[0]

def condition(tempDr, tempndr):
    return tempDr > tempndr

def true_fn(tempDr, tempndr):
    return 1;

def false_fn(tempDr, tempndr):
    return 0;

with tf.compat.v1.Session() as sess:
    for i in range(sess.run(num_samples)):
        tempDr = prediction[i][0]
        tempndr = prediction[i][1]
        result = true_fn(tempDr, tempndr)
        output = tf.cond(condition(tempDr, tempndr), lambda: true_fn(tempDr, tempndr), lambda: false_fn(tempDr, tempndr))
        if result==1:
            print('Diabetic Retinopathy')
        else:
            print('No Diabetic Retinopathy')

image.show()
```

Diabetic Retinopathy

```
6s import tensorflow as tf

# assuming `prediction` is a symbolic tensor
image = Image.open('/content/drive/MyDrive/Colab_Notebooks/ml_project/test/nodr/46_right.jpeg')
image.show()
processed_image = preprocess_image(image, target_size=(224, 224))
prediction = model.predict_on_batch(processed_image)
prediction_shape = tf.shape(prediction)
num_samples = prediction_shape[0]

def condition(tempDr, tempndr):
    return tempDr > tempndr

def true_fn(tempDr, tempndr):
    return 1;

def false_fn(tempDr, tempndr):
    return 0;

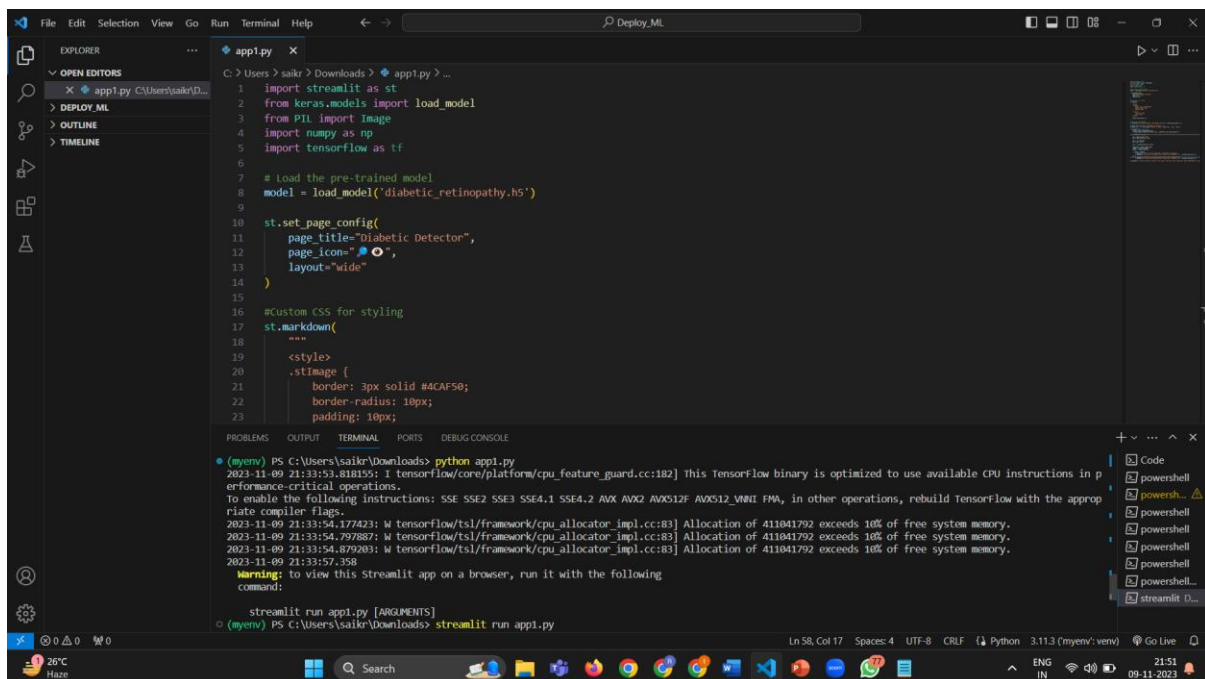
with tf.compat.v1.Session() as sess:
    for i in range(sess.run(num_samples)):
        tempDr = prediction[i][0]
        tempndr = prediction[i][1]
        result = false_fn(tempDr, tempndr)
        output = tf.cond(condition(tempDr, tempndr), lambda: true_fn(tempDr, tempndr), lambda: false_fn(tempDr, tempndr))
        if result==1:
            print('Diabetic Retinopathy')
        else:
            print('No Diabetic Retinopathy')
```

No Diabetic Retinopathy

## DEPLOYMENT:

In the final phase of the project, the developed automated prediction model for Diabetic Retinopathy, powered by Convolutional Neural Networks (CNN), was ready for deployment. To make this invaluable tool accessible to healthcare professionals and patients alike, we opted for the user-friendly and interactive platform offered by Streamlit. Streamlit provided an efficient and elegant means of deploying the model, allowing users to upload retinal images effortlessly and receive real-time predictions.

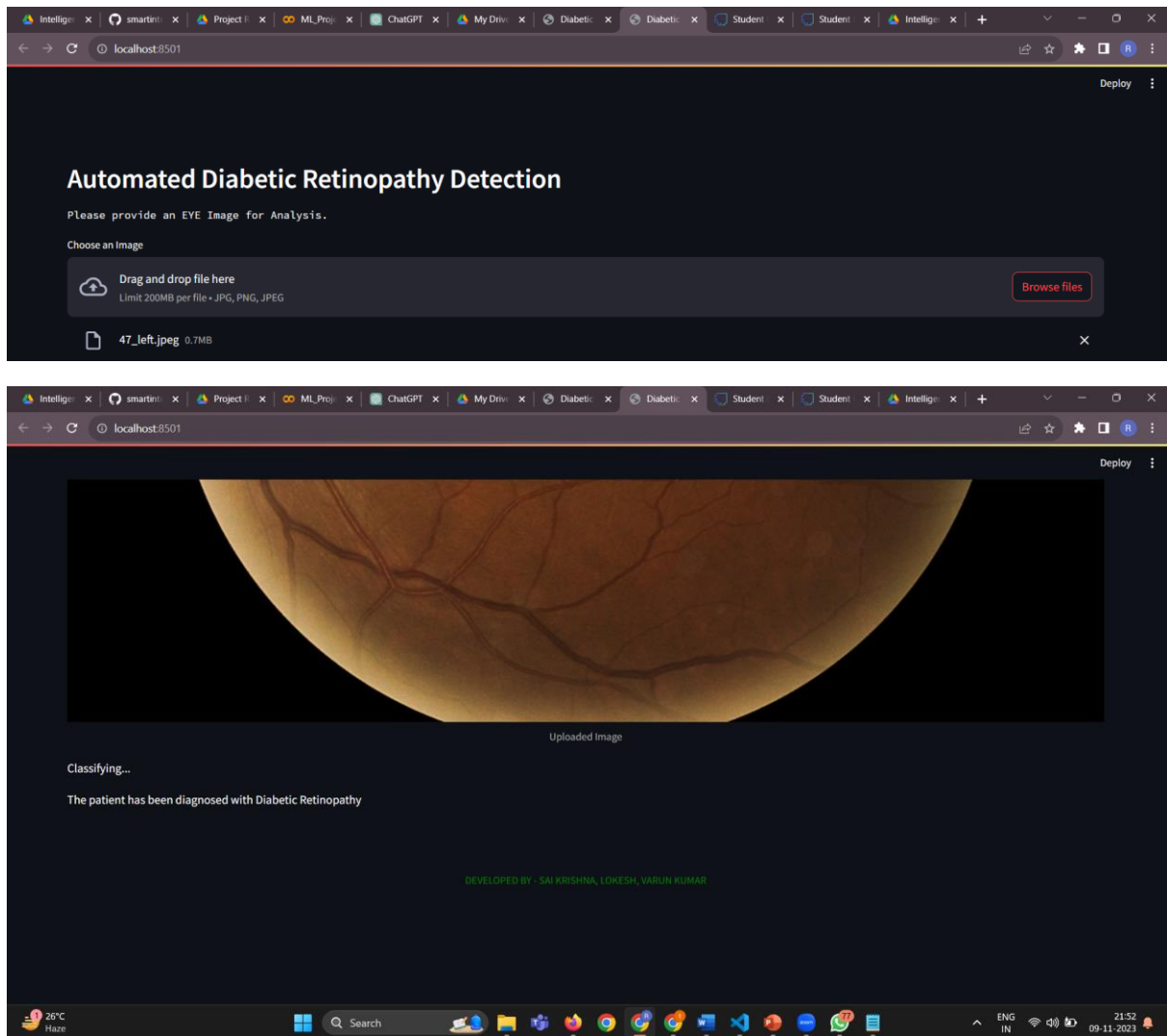
The model, saved in the .h5 format, was seamlessly integrated into the Streamlit application. This format, known for its compatibility with deep learning models, ensured that the model's architecture, weights, and training history were preserved during deployment. Streamlit's simplicity and flexibility enabled the creation of an intuitive user interface, where users could easily upload their retinal images, receive immediate predictions, and access interpretability tools to understand the model's decisions.



```
File Edit Selection View Go Run Terminal Help
Deploy_ML

EXPLORER
  OPEN EDITORS
    app1.py C:\Users\sakr\Downloads
  DEPLOY_ML
  OUTLINE
  TIMELINE

C:\Users\sakr\Downloads> app1.py
1 import streamlit as st
2 from keras.models import load_model
3 from PIL import Image
4 import numpy as np
5 import tensorflow as tf
6
7 # Load the pre-trained model
8 model = load_model('diabetic_retinopathy.h5')
9
10 st.set_page_config(
11     page_title="Diabetic Detector",
12     page_icon="🩺",
13     layout="wide"
14 )
15
16 #Custom CSS for styling
17 st.markdown(
18     """
19     <style>
20     .stImage {
21         border: 3px solid #4CAF50;
22         border-radius: 10px;
23         padding: 10px;
24     }
25     """)
26
27 PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
28 (myenv) PS C:\Users\sakr\Downloads> python app1.py
29 2023-11-09 21:33:53.818155: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in p
30 erformance-critical operations.
31 To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVXS12F AVXS12_VNNI FMA, in other operations, rebuild TensorFlow with the appro
32 priate compiler flags.
33 2023-11-09 21:33:54.177423: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 411841792 exceeds 10% of free system memory.
34 2023-11-09 21:33:54.797887: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 411841792 exceeds 10% of free system memory.
35 2023-11-09 21:33:54.879203: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 411841792 exceeds 10% of free system memory.
36 2023-11-09 21:33:57.358
37 Warning: to view this Streamlit app on a browser, run it with the following
38 command:
39
40 streamlit run app1.py [ARGUMENTS]
41 (myenv) PS C:\Users\sakr\Downloads> streamlit run app1.py
```



# **TESTING**

The purpose of testing can be quality assurance, verification and evidence, or responsibility estimation. Testing can be used as a general metric as well. Correctness testing and responsibility testing are two major areas of testing. Software testing is a trade- off between budget, time and quality. The main course of testing is to check for the actuality of scars or crimes in a program or design or product, predicated up on some predefined instructions or conditions.

Verification Checks consistence with the i/ p's

The process of determining whether or not the products of a given phase of the software development cycle meets the performance way and can be traced to the incoming objects established during the former phase. The ways for verification are testing, examination.

evidence Checks consistence with the user Conditions.

The process of assessing software at the end of the software development process to ensure compliance with software conditions. The ways for evidence are testing, examination and reviewing. Software Testing is an empirical exploration conducted to give stakeholders with information about the quality of the product or service under test, with respect to the terrain in which it's intended to operate. This includes, but is not limited to, the process of executing a program or operation with the intent of finding software bugs.

Testing can no way completely establish the correctness of computer software. rather, it furnishes a review or comparison that compares the state and behavior of the product against oracles — principles or mechanisms by which someone might recognize a problem. These oracles may include( but are not limited to) specifications, analogous products, formerly performances of the same product, consequences about intended or anticipated purpose, user or customer prospects, applicable morals, applicable laws, or other criteria.

Over its actuality, computer software has continued to grow in complexity and size. Every software product has a target cult. For illustration, the cult for video game software is completely different from banking software. therefore, when an association develops or differently invests in a software product, it presumably must assess whether the software product will be respectable to its end stoners, its target cult, its purchasers, and other stakeholders. Software testing is the process of trying to make this assessment.

### Testcase-1

```
with graph.as_default():
    image = Image.open(r"/content/drive/MyDrive/Colab Notebooks/dataset/30_left.jpeg")
    image.show()
    processed_image = preprocess_image(image, target_size=(224, 224))
    prediction = model.predict(processed_image).tolist()

    for i in range(0,int(len(prediction))):
        tempDr = prediction[i][0]
        tempnDr = prediction[i][1]
        if tempDr>tempnDr:
            print("Diabetics")
        elif tempDr<tempnDr:
            print("NO Diabetics")
```

Diabetics

- For the above code

Input data:

/content/drive/MyDrive/colab Notebooks/30\_left.jpeg

Output data:

Diabetics

### TestCase-2

```
with graph.as_default():
    image = Image.open(r"/content/drive/MyDrive/Colab Notebooks/dataset/1002_left.jpeg")
    image.show()
    processed_image = preprocess_image(image, target_size=(224, 224))
    prediction = model.predict(processed_image).tolist()

    for i in range(0,int(len(prediction))):
        tempDr = prediction[i][0]
        tempnDr = prediction[i][1]
        if tempDr>tempnDr:
            print("Diabetics")
        elif tempDr<tempnDr:
            print("NO Diabetics")
```

NO Diabetics

- For the above code

Input data:

/content/drive/MyDrive/colab Notebooks/1002\_left.jpeg

Output data:

No Diabetics

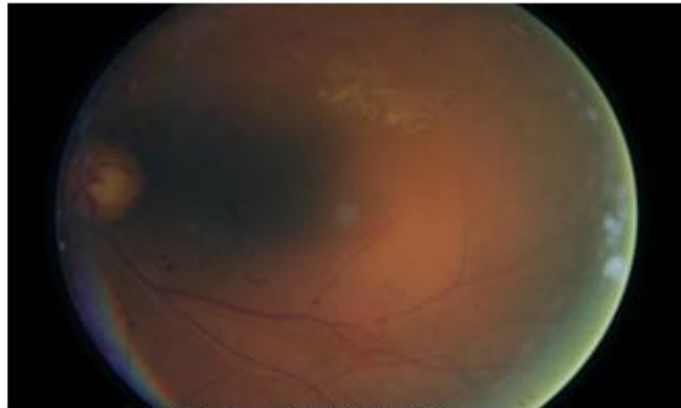


Fig 12. Eye with Diabetic Retinopathy

The above Fig 12 is fundus image of Eye with Diabetic Retinopathy  
/content/drive/MyDrive/colab Notebooks/30\_left.jpeg



Fig 13. Normal Eye

The above Fig 13 is fundus image of Normal Eye.  
/content/drive/MyDrive/colab Notebooks/1002\_left.jpeg

## **ADVANTAGES & DISADVANTAGES**

### **Advantages:**

**Early Detection:** A well-trained Diabetic Retinopathy model can detect the disease at an early stage, allowing for timely intervention and treatment. This can prevent or reduce the severity of vision loss in diabetic patients.

**Scalability:** Once deployed, the model can scale to analyze a large number of retinal images efficiently, making it suitable for screening programs and healthcare facilities with a high patient load.

**Consistency:** Machine learning models provide consistent and objective assessments, reducing variability in human interpretations of retinal images.

**Quick Results:** Model predictions can be generated rapidly, enabling quicker diagnosis and treatment planning.

**Reduction of Healthcare Costs:** Early detection and intervention can lead to reduced healthcare costs by preventing severe cases that require expensive treatments.

**Accessibility:** The model can be accessed remotely, which is particularly useful in underserved or remote areas where access to specialized eye care is limited.

### **Disadvantages:**

**Data Quality:** The performance of the model heavily depends on the quality and diversity of the training data. Poor or biased training data can lead to inaccurate predictions.

**Interpretability:** Deep learning models, such as convolutional neural networks, can be challenging to interpret, making it difficult to understand the basis for their predictions.

**Hardware and Infrastructure:** Training and deploying deep learning models may require substantial computational resources and infrastructure.

**Ethical Concerns:** Ensuring fairness, transparency, and ethical use of AI in healthcare is a critical concern. Bias in the data or algorithms can lead to unequal access to care.

## **CONCLUSION**

In conclusion, the development of a Diabetic Retinopathy classification model represents a pivotal intersection of cutting-edge technologies and healthcare innovation. Leveraging the power of machine learning, particularly deep learning with Convolutional Neural Networks (CNNs), this model empowers healthcare professionals to harness the insights hidden within retinal images. Computer vision technologies play a pivotal role in the preprocessing and analysis of images, ensuring that even subtle signs of Diabetic Retinopathy are not overlooked. These advancements bring us closer to the goal of early detection and intervention, ultimately enhancing patient outcomes.

The importance of data management and databases cannot be overstated in this endeavor. Technologies for efficient data organization and storage are instrumental in managing vast repositories of retinal images and patient information. Cloud computing resources provide the necessary infrastructure for training and deploying the model, enabling remote access and scalability. This technological infrastructure offers a paradigm shift in healthcare, making the model accessible to underserved regions and healthcare facilities with limited resources.

APIs and web services facilitate the integration of the model into clinical workflows and user interfaces, allowing healthcare professionals to seamlessly incorporate its capabilities into their diagnostic processes. This technological synergy marks a significant step forward in the quest for precision medicine and personalized care. It is crucial to recognize that, while the model holds immense potential, its success relies on the ethical use of technology, data privacy, and adherence to regulatory standards. As we navigate this evolving landscape, we must continue to harness the power of technology responsibly, ensuring that the benefits of the Diabetic Retinopathy classification model are accessible to all and that it aligns with the highest standards of healthcare excellence.



## **FUTURE SCOPE**

**Improved Accuracy and Robustness:** Future advancements will focus on enhancing the accuracy and robustness of Diabetic Retinopathy classification models. This includes refining the model's ability to detect subtle changes in retinal images and reducing false positives and false negatives. Advanced deep learning architectures and novel data augmentation techniques will contribute to improved performance.

**Interpretable AI:** As the adoption of artificial intelligence (AI) in healthcare grows, there will be a greater emphasis on making AI models, including those for Diabetic Retinopathy, more interpretable. Explainable AI techniques will help healthcare professionals better understand the model's decisions, leading to increased trust and adoption.

**Personalized Medicine:** The future holds the potential for personalized treatment plans based on an individual's risk profile for Diabetic Retinopathy. Machine learning models can be used to tailor interventions and follow-up schedules, optimizing healthcare resources and improving patient outcomes.

**Telemedicine and Remote Screening:** Telemedicine and remote screening programs will expand, driven by the accessibility and scalability of Diabetic Retinopathy classification models. Patients in remote or underserved areas will benefit from access to retinal screening without the need for in-person visits.

**Multi-Modal Imaging:** Integrating multiple imaging modalities, such as optical coherence tomography (OCT) and fundus photography, will provide a more comprehensive view of retinal health. Future models will combine these modalities for a more accurate diagnosis.

**Real-time Monitoring:** Continuous monitoring of retinal health will become more feasible, allowing for early detection of changes and immediate intervention. Wearable devices and smartphone applications may play a role in this real-time monitoring.

**Global Adoption:** The global adoption of Diabetic Retinopathy classification models will lead to standardized screening and diagnostic processes. This harmonization of healthcare practices will benefit patients worldwide.

## **APPENDIX**

**Codes and documents:**

[https://drive.google.com/drive/folders/1bc9w8\\_qH\\_vAwrQRAweHIJ1s8ig4WtV2h?usp=drive\\_link](https://drive.google.com/drive/folders/1bc9w8_qH_vAwrQRAweHIJ1s8ig4WtV2h?usp=drive_link)

**Video presentation:**

[https://drive.google.com/file/d/1uYdDsRWL9P5-RenIfejNZH3XfLrnO-ec/view?usp=drive\\_link](https://drive.google.com/file/d/1uYdDsRWL9P5-RenIfejNZH3XfLrnO-ec/view?usp=drive_link)